Modelling and verification of probabilistic systems

Marta Kwiatkowska
School of Computer Science
THE UNIVERSITY OF BIRMINGHAM
www.cs.bham.ac.uk/~mzk
www.cs.bham.ac.uk/~dxp/prism

Lucent, 10th November 2004

Overview

- Motivation
- Probabilistic model checking
  - The models
  - Specification languages
  - What does it involve?
  - The PRISM model checker
- Case studies
  - Self-stabilisation
  - Dynamic power management
  - IPv4 Zeroconf dynamic configuration protocol
  - Root contention in IEEE 1394 FireWire
- Challenges for future

The future: ubiquitous computing

Mobile, wearable, wireless devices (WiFi, Bluetooth)
Ad hoc, dynamic, ubiquitous computing environment
Security, privacy, anonymity protection on the Internet
Self-configurable - no need for men/women in white coats!
Fast, responsive, power efficient, …

Probability helps

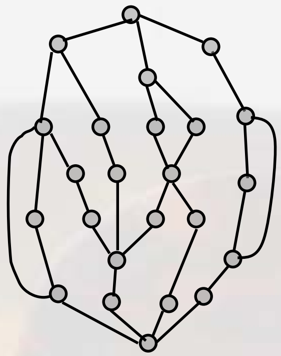
- In distributed co-ordination algorithms
  - As a symmetry breaker
    - "leader election is eventually resolved with probability 1"
  - In gossip-based routing and multicasting
    - "the message will be delivered to all nodes with high probability"

- When modelling uncertainty in the environment
  - To quantify failures, express soft deadlines, QoS
    - "the chance of shutdown is at most 0.1%"
    - "the probability of a frame delivered within 5ms is at least 0.91"
  - To quantify environmental factors in decision support
    - "the expected cost of reaching the goal is 100"

- When analysing system performance
  - To quantify arrivals, service, etc, characteristics
    - "in the long run, mean waiting time in a lift queue is 30 sec"

Verification via model checking…

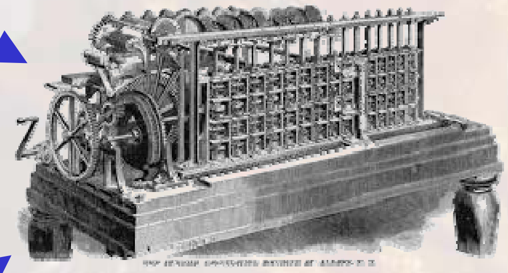
or falsification?



The model

send $\rightarrow \Diamond$deliver

Temporal logic specification



Model Checker

or

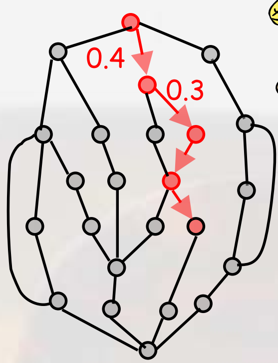
Error trace

Line 5: …
Line 21: …
Line 15: …
…
Line 27: …
Line 45: …

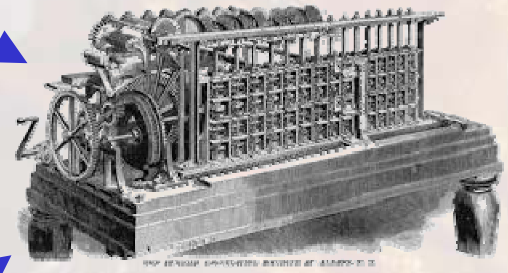
Also refinement checking, equivalence checking, …

Probabilistic model checking...

in a nutshell



Probabilistic model



Probabilistic Model Checker

or

or

The probability

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
…
State 12: 0
State 13: 0.1245

send $\rightarrow \mathbf{P}_{\geq 0.9}(\Diamond$deliver$)$

Probabilistic temporal logic specification

Probability elsewhere

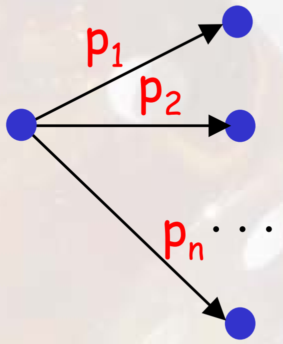
- In performance modelling
  - Pioneered by Erlang, in telecommunications, ca 1910
  - Models: typically continuous time Markov chains
  - Emphasis on steady-state and transient probabilities

- In stochastic planning
  - Cf Bellman equations, ca 1950s
  - Models: Markov decision processes
  - Emphasis on finding optimum policies

- Our focus, probabilistic model checking
  - Distinctive, on automated verification for probabilistic systems
  - Temporal logic specifications, automata-theoretic techniques
  - Shared models
  - Exchanging techniques with the other two areas

Probabilistic models: discrete time

- Labelled transition systems
  - Discrete time steps
  - Labelling with atomic propositions
- Probabilistic transitions
  - Move to state with given probability
  - Represented as discrete probability distribution
- Model types
  - Discrete time Markov chains (DTMCs): probabilistic choice only
  - Markov decision processes (MDPs): probabilistic choice and nondeterminism



$\sum_i p_i = 1$

Theory timeline: discrete models

Qualitative (with probability 1 or 0)

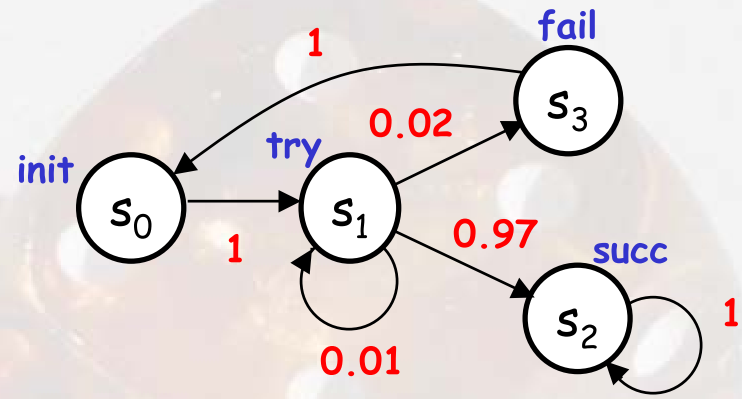
- 1983 Hart-Sharir-Pnueli
- 1985 Vardi
- 1988 Courcoubetis-Yannakakis

Quantitative (with arbitrary probability)

- 1991 Larsen-Skou (probab. bisimulation)
- 1994 Hansson-Jonsson (DTMC model checking)
- 1995 Bianco-de Alfaro (MDP model checking)
- 1995 Segala-Lynch (probab. simulation)
- 1997 Huth-Kwiatkowska [LICS] (probab. mu-calculus)
- 1997 Baier et al (DTMC model checking)
- 1998 Baier-Kwiatkowska (MDPs + fairness)
- 1999 Kwiatkowska-Norman-Segala-Sproston (PTAs)
- 2001 Kwiatkowska-Norman-Sproston (infinite state)

Discrete-Time Markov Chains (DTMCs)

- Features:
  - Only probabilistic choice in each state



- Formally, (S, s_0, P, L):
  - S finite set of states
  - s_0 initial state
  - P: S £ S ! [0,1] probability matrix, s.t. $\sum_{s'} P(s,s') = 1$, all s
  - L: S ! 2^{AP} atomic propositions

- Unfold into infinite paths $s_0 s_1 s_2 s_3 s_4 \ldots$ s.t. $P(s_i, s_{i+1}) > 0$, all i

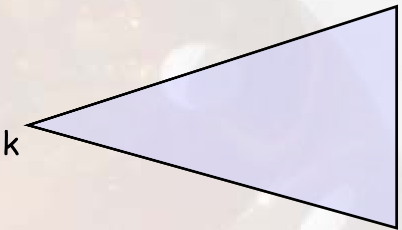
- Probability for finite paths, multiply along path

  e.g. $s_0\, s_1\, s_1\, s_2$ is 1 ¢ 0.01 ¢ 0.97 = 0.0097

Probability space

- Intuitively:
  - Sample space = infinite paths $Path_s$ from s
  - Event = set of paths
  - Basic event = cone

$ss_1s_2 \ldots s_k$



- Formally, ($Path_s$, Ω, Pr)
  - For finite path $\omega = ss_1 \ldots s_n$, define probability

$$P(\omega) = \begin{cases} 1 \text{ if } \omega \text{ has length one} \\ P(s,s_1) ¢ \ldots ¢ P(s_{n-1},s_n) \text{ otherwise} \end{cases}$$

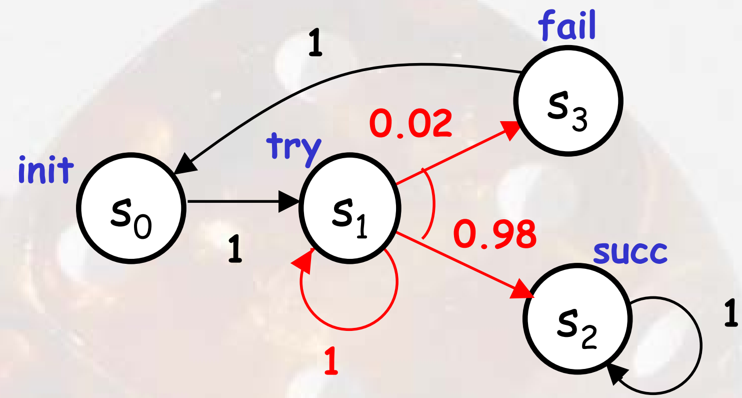
  - Take Ω least σ-algebra containing cones

$$C(\omega) = \{ \pi \; 2 \; Path_s \mid \omega \text{ is prefix of } \pi \}$$

  - Define $Pr(C(\omega)) = \mathbf{P}(\omega)$, all ω
  - Pr extends uniquely to measure on $Path_s$

Markov Decision Processes (MDPs)

- Features:
  - Nondeterministic choice
  - Parallel composition of DTMCs



- Formally, $(S, s_0, Steps, L)$:
  - S finite set of states
  - s_0 initial state
  - Steps maps states s to sets of probability distributions μ over S
  - L: S ! 2^{AP} atomic propositions

- Unfold into infinite paths $s_0 \mu_0 s_1 \mu_1 s_2 \mu_2 s_3 \ldots$ s.t. $\mu_i(s_i, s_{i+1}) > 0$, all i

- Probability space induced on $Path_s$ by adversary (policy) A mapping finite path $s_0 \mu_0 s_1 \mu_1 \ldots s_n$ to a distribution from state s_n

The logic PCTL: syntax

- Probabilistic Computation Tree Logic [HJ94,BdA95,BK98]
  - For DTMCs/MDPs
  - New probabilistic operator, e.g. send $\rightarrow$ $\mathbf{P}_{\geq 0.9}(\Diamond \text{deliver})$

    "whenever a message is sent, the probability that it is eventually delivered is at least 0.9"

- The syntax of state and path formulas of PCTL is:

$$\phi ::= \text{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \mathbf{P}_{\bowtie p}(\alpha)$$
$$\alpha ::= X\,\phi \mid \phi\,U\,\phi$$

where $p \in [0,1]$ is a probability bound and $\bowtie \in \{ <, >, \ldots \}$

- Subsumes the qualitative variants [Var85,CY95] $\mathbf{P}_{=1}(\alpha)$, $\mathbf{P}_{>0}(\alpha)$

- Extension with cost/rewards and expectation operator $\mathbf{E}_{\bowtie c}(\phi)$

The logic PCTL: semantics

- Semantics is parameterised by a class of adversaries Adv
  - "under any scheduling, the probability bound is true at state s"
  - reasoning about worst-case/best-case scenario

- The probabilistic operator is a quantitative analogue of 8, 9

$s \;^{2}{}_{Adv}\; \mathbf{P}_{» p}(\alpha)$, $Pr^{A}\{\, \pi \; 2 \; Path^{A}_{s} \; j \; \pi \;^{2}{}_{Adv}\; \alpha \,\} » p$
for all A 2 Adv

PCTL semantics: summary

- Semantics of state formulas:

$s \ {}^2{}_{Adv}\ a \quad , \quad a\ 2\ L(s)$

$s \ {}^2{}_{Adv}\ :\phi \quad , \quad s \ {}^2{}_{Adv}\ \phi$

$s \ {}^2{}_{Adv}\ \phi_1\ Æ\ \phi_2 \quad , \quad s \ {}^2{}_{Adv}\ \phi_1$ and $s \ {}^2{}_{Adv}\ \phi_2$

- Semantics of path formulas:

$\pi \ {}^2{}_{Adv}\ X\ \phi \quad , \quad \pi = s_0 \cdots$ and $s_1 \ {}^2{}_{Adv}\ \phi$

$\pi \ {}^2{}_{Adv}\ \phi_1\ U\ \phi_2 \quad , \quad \pi = s_0 \cdots$ and 9 k s.t.
$s_k \ {}^2{}_{Adv}\ \phi_2$ and $8\ j \ ‹\ k\ .\ s_j \ {}^2{}_{Adv}\ \phi_1$

- The probabilistic operator:

$s \ {}^2{}_{Adv}\ \mathbf{P}_{» p}(\alpha) \quad , \quad Pr^A \{\ \pi\ 2\ Path^A{}_s\ j\ \pi \ {}^2{}_{Adv}\ \alpha\ \}\ »\ p$
for all A 2 Adv

The logic PCTL: model checking

- By induction on structure of formula, as for CTL

- For the probabilistic operator and Until, solve
  - recursive linear equation for DTMCs
  - linear optimisation problem (form of Bellman equation) for MDPs
  - typically iterative solution methods

- Need to combine
  - conventional graph traversal
  - numerical linear algebra and linear optimisation (value iteration)

- Qualitative properties (probability 1, 0) proceed by graph traversal [Var85,dAKNP97]

PCTL model checking for DTMCs

- By induction on structure of formula
- For the probabilistic operator
  - $Sat(\ P_{» p}(X\ \phi)\ )\quad ,\quad \{s\ 2\ S \mid \sum_{s'\ 2\ Sat(\phi)} P(s,s')\ »\ p\}$
  - $Sat(\ P_{» p}(\phi_1\ U\ \phi_2)\ )\ ,\quad \{s\ 2\ S \mid x_s\ »\ p\}$

    where x_s, s 2 S, are obtained from the recursive linear equation

$$
x_s \;=\; \begin{cases} 0 & \text{if } s\ 2\ S^{no} \\ 1 & \text{if } s\ 2\ S^{yes} \\ \sum_{s'\ 2\ S} P(s,s')\ ¢\ x_{s'} & \text{if } s\ 2\ S n (S^{no} [\ S^{yes}) \end{cases}
$$

and

S^{yes} – states that satisfy $\phi_1\ U\ \phi_2$ with probability exactly 1

S^{no} – states that satisfy $\phi_1\ U\ \phi_2$ with probability exactly 0

PCTL model checking for DTMCs

- For the remaining formulas standard:

$$\begin{aligned} Sat(a) &= L(a) \\ Sat(\neg\phi) &= S \setminus Sat(\phi) \\ Sat(\phi_1 \wedge \phi_2) &= Sat(\phi_1) \cap Sat(\phi_2) \end{aligned}$$

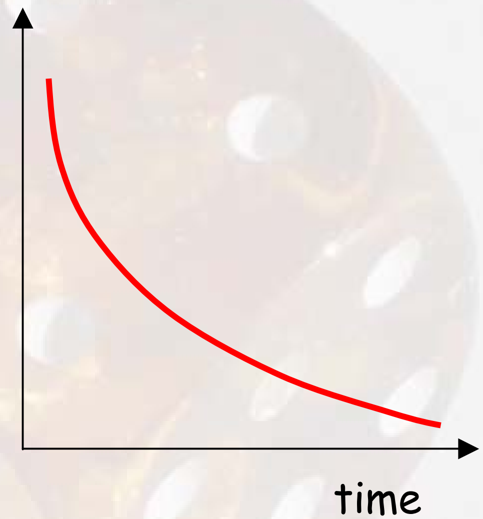
- S^{yes}, S^{no} can be precomputed by graph traversal [Var85] (or BDD fixed point computation)

- Need to combine
  - Conventional graph-theoretic traversal
  - Numerical linear algebra

Probabilistic models: continuous

- Assumptions on time and probability
  - Continuous passage of time
  - Continuous randomly distributed delays
  - Continuous space

- Model types
  - Continuous time Markov chains (CTMCs): exponentially distributed delays, discrete space, no nondeterminism
  - Probabilistic Timed Automata (PTAs): dense time, (usually) discrete probability, admit nondeterminism
  - (not considered) Labelled Markov Processes (LMPs): continuous space/time, no nondeterminism



$S_0^{+1} f(x)dx = 1$

Theory timeline: continuous models

Continuous distributions

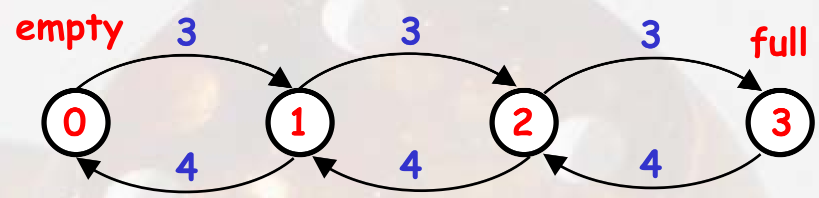
- 1991 Alur-Courcoubetis-Dill (GSMPs)
- 1996 Aziz-Sanwal-Singhal-Brayton (logic CSL)
- 1998 de Alfaro (long-run average)
- 1999 Baier, Katoen, Hermanns (CTMC model checking)
- 2000 Baier, Haverkort, Hermanns, Katoen (uniformis.)
- 2000 Kwiatkowska-Norman-Segala-Sproston (cont. PTAs)

Continuous space, approximation

- 1997 Blute-Desharnais-Edalat-Panangaden [LICS] (bisim. LMPs)
- 1998 Desharnais-Edalat-Panangaden (logic LMPs)
- 1999 Desharnais-Gupta-Jagadeesan-Panangaden [CONCUR] (metric)
- 2000 Desharnais-Gupta-Jagadeesan-Panangaden [LICS] (approx. LMPs)

Continuous Time Markov Chains (CTMCs)

- Features:
  - Discrete states and real time
  - Exponentially distributed random delays



- Formally:
  - Set of states S plus rates $R(s,s') > 0$ of moving from s to s'
  - Probability of moving from s to s' by time $t > 0$ is $1 - e^{-R(s,s') \cdot t}$
  - Transition rate matrix $S \times S \to \mathbb{R}_{\geq 0}$
- Unfold into infinite paths $s_0 t_0 s_1 t_1 s_2 t_2 s_3 \ldots$
  - $prob_s(s')$, probability of being in s' in the long-run, starting in s
  - $prob_s(s',t)$, probability of being in s' at time instant t
- But: no nondeterminism

The logic CSL: syntax

- Continuous Stochastic Logic [ASSB96,BKH99]
  - For CTMCs, based on PCTL, for example
    - $P_{‹\ 0.85}$(}$^{‹15}$ full), probability operator
      "the probability of queue becoming full within 15 secs is ‹ 0.85"
    - $S_{‹\ 0.01}$(down), steady-state operator
      "in the long run, the probability the system is down is less than 1%"

- The syntax of state and path formulas of CSL is:

  φ ::= true | a | φ Æ φ | :φ | **S**$_{»\ p}$(φ) | **P**$_{»\ p}$(α)
  α ::= X φ | φ U$^{·\ t}$ φ | φ U φ

  where p 2 [0,1] is a probability bound, t 2 R$_{¸0}$ and » 2 { ‹, ›, … }

- Extension with time intervals for until, cost/rewards and expectation operator **E**$_{»\ c}$(φ)

CSL semantics

- Semantics of bounded until:

  $\pi\ ^{2}\ \phi_1\ U^{\cdot\ t}\ \phi_2$, iff ϕ_2 satisfied at time instant t along $\pi = s_0 \cdots$ and ϕ_1 satisfied at all preceding time instants

- The added operators:

  $s\ ^{2}\ \mathbf{S}_{»\ p}(\phi)$, $\Sigma_{s'\ ^{2}\ \phi}\ prob_s\ (s')\ »\ p$ where $prob_s\ (s')$ is prob. of being in s' in the long-run, having started in s

  $s\ ^{2}\ \mathbf{P}_{»\ p}(\alpha)$, $Pr\ \{\ \pi\ 2\ Path_s\ j\ \pi\ ^{2}\ \alpha\ \}\ »\ p$ where Pr is probability measure on paths as for PCTL

- Semantics of remaining formulas as for PCTL

The logic CSL: model checking

- By induction on structure of formula, as for PCTL except for
  - $\mathbf{S}_{» p}(\phi)$ and $\mathbf{P}_{» p}(\phi_1 \; U^{\cdot \, t} \; \phi_2)$
- The steady-state operator
  - Requires computation of steady-state probabilities
  - Reduces to graph traversal and (iterative) solution of linear equation system
- The time-bounded until
  - Reduces to transient analysis
  - Transform CTMC by removing all outgoing transitions from states satisfying ϕ_2 or $:\phi_1$
  - Then $Pr\,\{\, \pi\ 2\ Path_s \ j\ \pi\ ^2\ \phi\ U^{\cdot\, t}\ \phi\ \} = \Sigma_{s'\ ^2\ \phi 2}\, prob_s\,(s',t)$
  - Computed by using uniformisation
  - More efficient and stable, iterative computation

Probabilistic model checking in practice

- Model construction: probability matrices
  - Enumerative
    - Manipulation of individual states
    - Size of state space main limitation
  - Symbolic
    - Manipulation of sets of states
    - Compact representation possible in case of regularity
- Temporal logic model checking: currently limited to
  - discrete probability/space models
  - CTMCs
  - Simulation admits more general distributions
- Probabilistic Symbolic Model Checker PRISM

The PRISM tool: overview

- Functionality
  - Direct support for models: DTMCs, MDPs and CTMCs
  - Extension with costs/rewards, expectation operator
  - PTAs with digital clocks by manual translation
  - Connection from KRONOS to PRISM for PTAs
  - Experimental implementation using DBMs/DDDs for PTAs

- Input languages
  - System description
    - probabilistic extension of reactive modules [Alur and Henzinger]
  - Probabilistic temporal logics: PCTL and CSL

- Implementation
  - Symbolic model construction (MTBDDs), uses CUDD [Somenzi]
  - Three numerical computation engines
  - Written in Java and C++

The PRISM tool: implementation

- Numerical engines
  - Symbolic, MTBDD based
    - Fast construction, reachability analysis
    - Very large models if regularity
  - Enumerative, sparse-matrix based
    - Generally fast numerical computation
    - Model size up to millions
  - Hybrid
    - Speed comparable to sparse matrices for numerical calculations
    - Limited by size of vector

- Experimental results
  - Several large scale examples: 10^{10} - 10^{30} states
  - No engine wins overall
  - See www.cs.bham.ac.uk/~dxp/prism

PRISM real-world case studies

- MDPs/DTMCs
  - Bluetooth device discovery [ISOLA'04]
  - Crowds anonymity protocol (by Shmatikov) [JSC 2003]
  - Randomised consensus [CAV'01,FORTE'02]
  - NAND multiplexing for nanotechnology (with Shukla) [VLSI'04]
  - Self-stabilising protocols
- CTMCs
  - Dynamic Power Management (with Shukla and Gupta) [HLDVT'02]
  - Dependability of embedded controller [INCOM'04]
- PTAs
  - IPv4 Zeroconf dynamic configuration [FORMATS'03]
  - Root contention in IEEE 1394 FireWire [FAC 2003, STTT 2004]
  - IEEE 802.11 (WiFi) Wireless LAN MAC protocol [PROBMIV'02]

PRISM Modelling Language

- Simple, state-based language for DTMCs/CTMCs/MDPs
  - based on Reactive Modules [Alur/Henzinger]
- Basic components:
  - modules (system components, parallel composition)
  - variables (finite-state, typed)
  - guarded commands (probabilistic, action-labelled)

[send] (s=2) -> p_{loss} : (s'=3)&(lost'=lost+1) + (1-p_{loss}) : (s'=4);

action guard probability update probability update

PRISM Modelling Language...

- Other features:
  - synchronisation on action labellings
  - process algebra style specifications
    - parallel composition: P1 ||| P2, P1 |[a,b]| P2, P1 || P2
    - action hiding/renaming: P/{a}, P{a<-b}
  - import of PEPA models
  - state-dependent probabilities/rates
  - global variables, macros, ...

PRISM Property Specifications

- Temporal logics: PCTL/CSL
  - probabilistic extensions of CTL
- Examples:
  - $P{\geq}1$ [true U terminate]
    "the algorithm eventually terminates successfully with probability 1"
  - $P{<}0.001$ [true $U{\leq}100$ error]
    "the probability of the system reaching an error state within 100 time units is less than 0.001"

PRISM Property Specifications

- More examples:
  - down => P>0.75 [!fail U[1,2.5] up]
    "when a shutdown occurs, the probability of system recovery being completed in between 1 and 2.5 hours, without further failures occurring, is greater than 0.75"
  - S<0.01 [num_sensors < min]
    "in the long-run, the probability that an inadequate number of sensors are operational is less than 0.01"
    (CSL only)

PRISM Property Specifications...

- Can write query formulae:
  - $P=?$ [true $U_{\leq 10}$ terminate]
    "what is the probability that the algorithm terminates successfully within 10 time units?"

- Can automate model checking with experiments:
  - $P=?$ [true $U_{\leq T}$ terminate]
    "what is the probability that the algorithm terminates successfully within time T?" for T=0,…,1000

Adding Costs/Rewards

- Augment states and transitions of model with real-valued rewards

- Instantaneous rewards
  - state-based
  - e.g. “queue size”, “concentration of reactant”
- Cumulative rewards
  - state- and transition-based
  - e.g. “time taken”, “power consumed”, “messages lost”

Properties - Instantaneous

- R=? [I=T]
  Expected reward at time instant T?

- R=? [S]
  Expected long-run reward?

Properties - Cumulative

- R=? [F A]
  Expected reward to reach A?

- R=? [C<=T]
  Expected reward by time T?

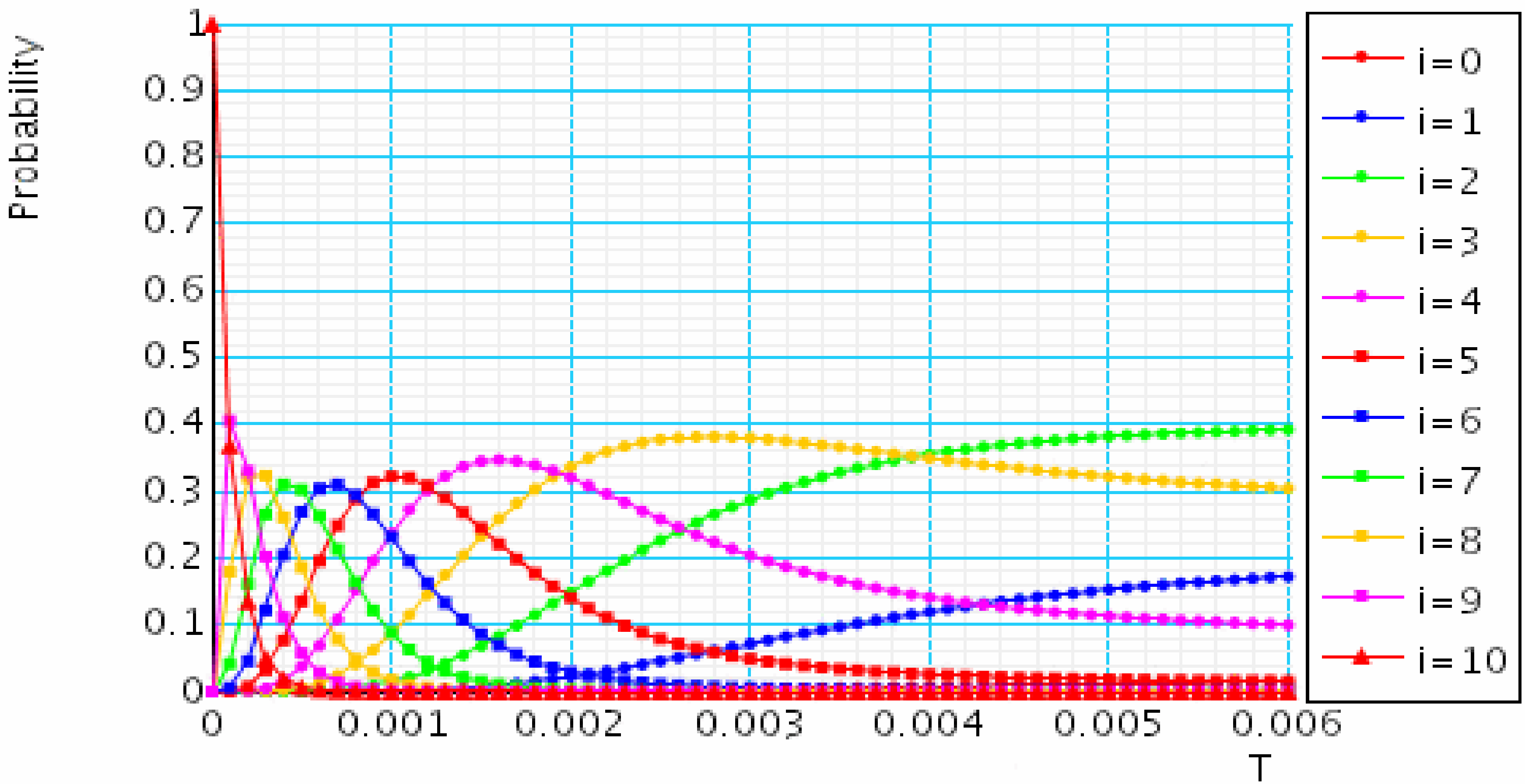
- R=? [S]
  Expected long-run reward per unit time?

Case Study: Molecular Reactions

- Time until a reaction occurs is given by an exponential distribution [Gillespie 1977]
  - model reactions using continuous time Markov chains
- Rate of reaction determined by:
  - base rate (empirically determined constant)
  - concentration of reactants (number of each type of molecule that takes part in the reaction)
- This case study: **$Na + Cl \leftrightarrow Na^{+} + Cl^{-}$**
  - forward base rate 100
  - backwards base rate 10
  - initially N1 Na molecules and N2 Cl molecules

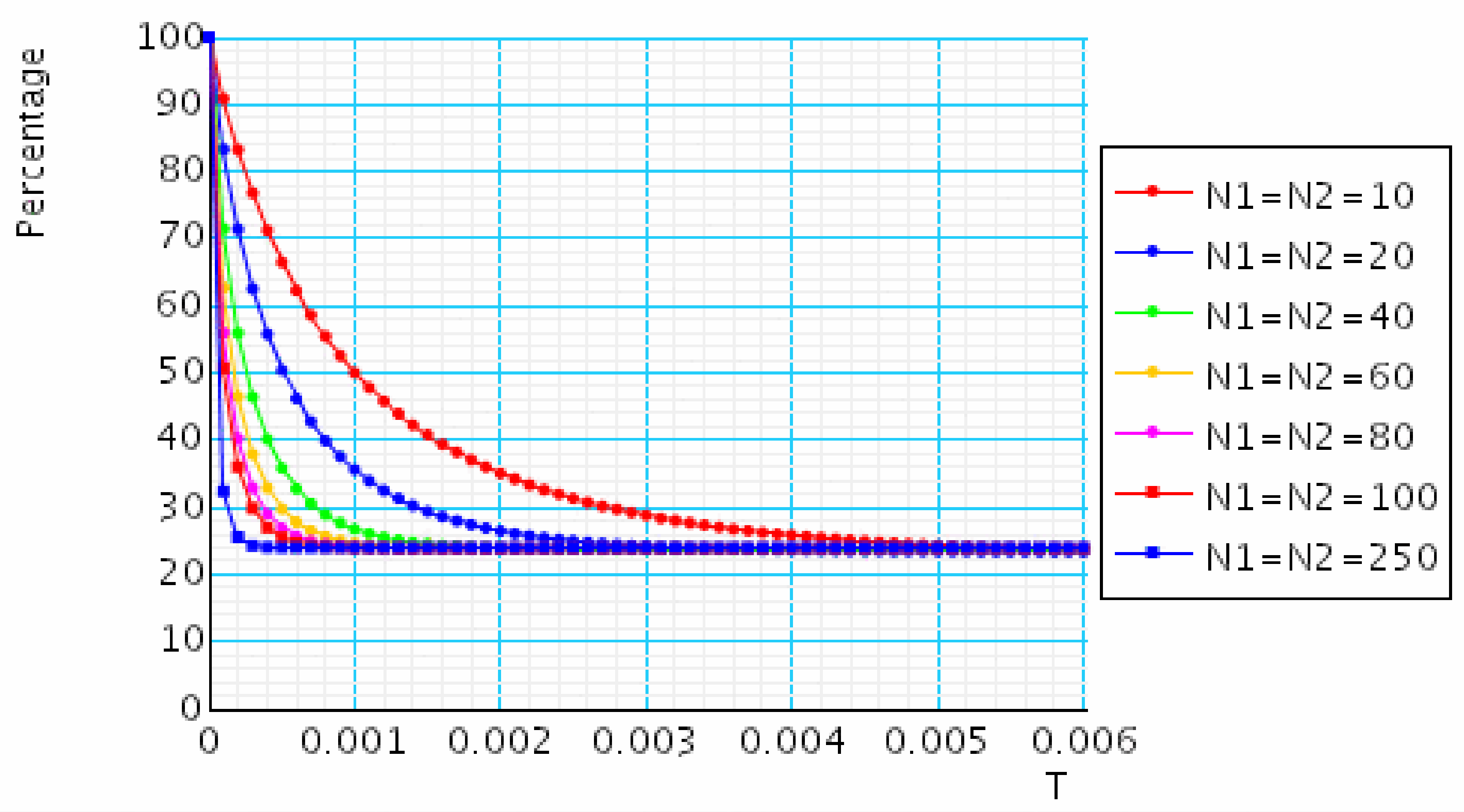
Results: Molecular Reactions

- $P_{=?}$ (true $U^{[T,T]}$ Na=i) 'probability i Na molecules at time T'



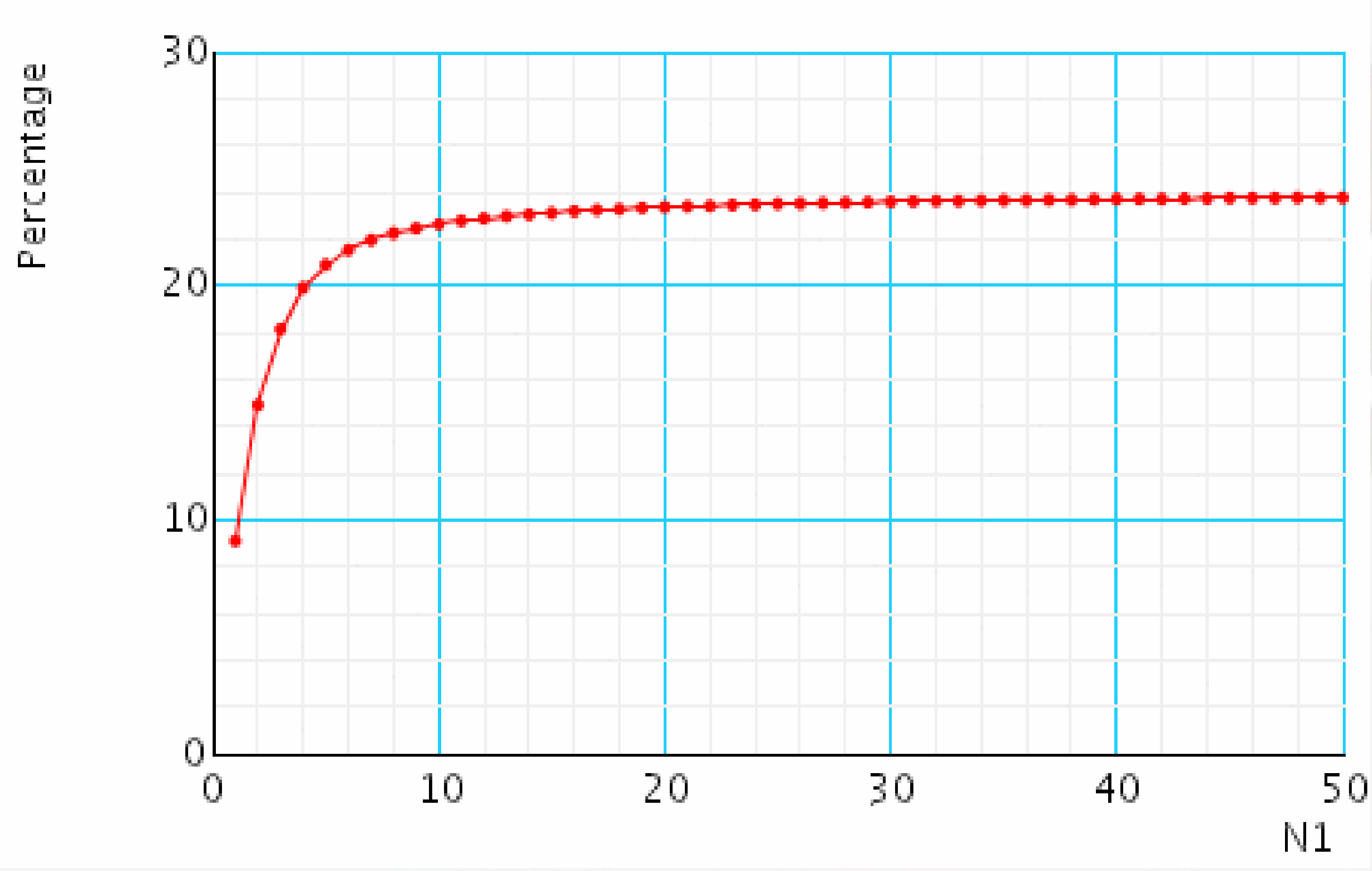
Results: Molecular Reactions

- $R_{=?}$ (I=T) 'expected percentage of Na molecules at time T'



Results: Molecular Reactions

- $\mathbf{R}_{=?}$ (S) 'expected percentage of Na molecules in the long run'



Case Study: Power management

- Power Management
  - controls power consumption in battery-operated devices
  - savings in power usage translate to extended battery life
  - important for portable, mobile and handheld electronic devices

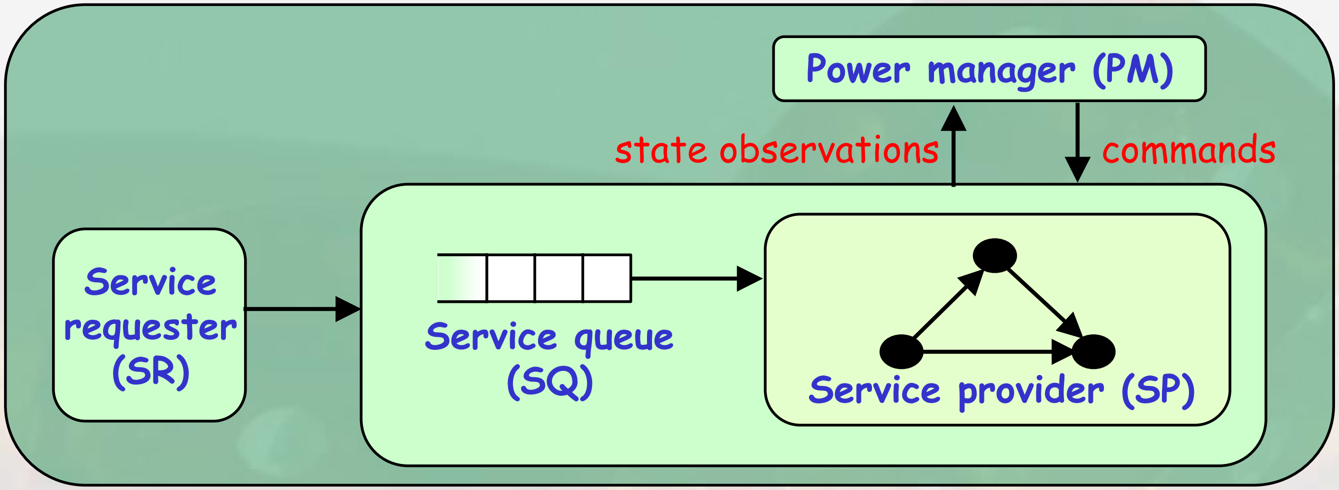
- System level power management
  - Manages various system devices for power optimisation
  - System components manufactured with several power modes e.g. disk drive has: active, idle, standby, sleep, …
  - Modes can be changed by the operating system through APIs
  - Exploits application characteristics
  - Needs to be implemented at the O/S level

Dynamic Power Management (DPM)

- DPM make optimal decisions at runtime based on:
  - Dynamically changing system state
  - Workload
  - Performance constraints

- Stochastic optimal control strategies for DPM
  - Construct a mathematical model of the system in PRISM
    - transition times modelled with exponential distributions
    - Model is CTMC or DTMC depending on time domain
  - Formulate stochastic optimisation problems
    - e.g. "optimise average energy usage while average delay below k"
  - Create stochastic strategies by solving optimisation problem
    - Exported to Maple for solution externally
  - Analyse optimal stochastic strategies directly in PRISM

DPM: The System Model



- Service requester (generates the service requests)
- Service provider (provides service to the requests)
- Service queue (buffers the requests)
- Power manager (monitors the states of the SR, SP and SQ and issues state-transition commands to the SP)

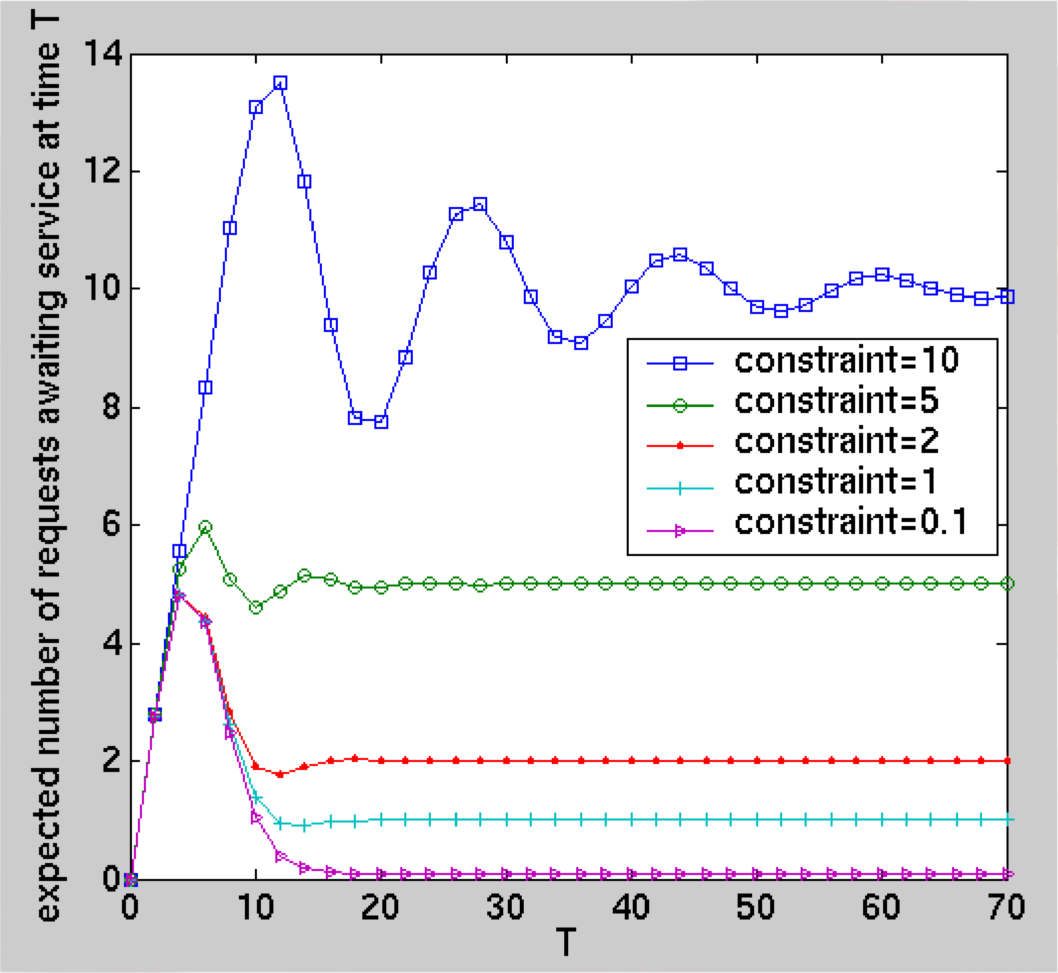
DPM Case Study: Fujitsu Disk Drive

- 4 state Fujitsu disk drive: busy, idle, standby and sleep)
  - Modelled as CTMC
- Policies:
  - Minimize: average power consumption
  - Constraint: average queue size

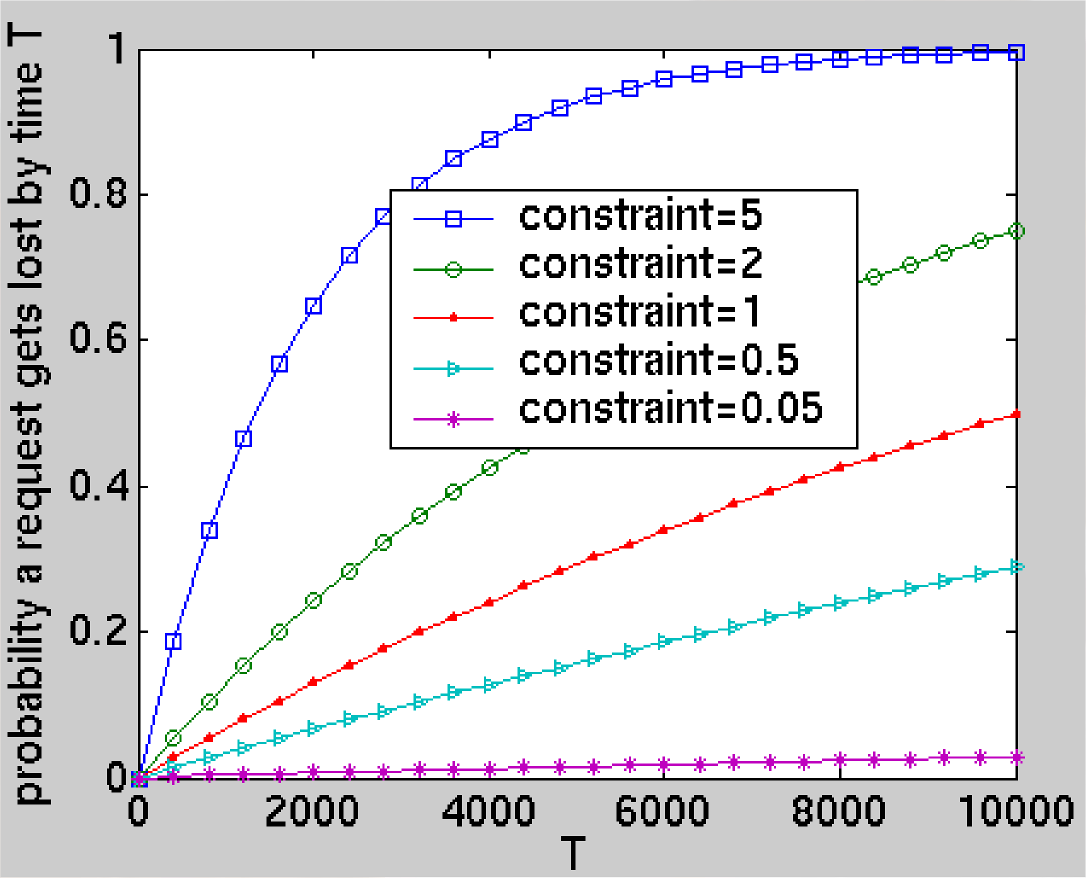
- Properties checked with PRISM:
  - Average power consumption/queue size
  - Average number of lost customers
  - Expected power consumption/queue size by time t
  - Expected number of lost customers by time t
  - Probability n requests lost by time t
  - Probability a request gets lost/served by time t

- See PRISM web site for further details

DPM Results: Fujitsu Disk Drive



DPM Results: Fujitsu Disk Drive



Case study: IPv4 Zeroconf protocol

- IPv4 ZeroConf protocol [Cheshire,Adoba,Guttman'02]
  - New IETF standard for dynamic network self-configuration
  - Link-local (no routers within the interface)
  - No need for an active DHCP server
  - Aimed at home networks, wireless ad-hoc networks, hand-held devices
  - "Plug and play"

- Self-configuration
  - Performs assignment of IP addresses
  - Symmetric, distributed protocol
  - Uses random choice and timing delays

IPv4 Zeroconf Standard

- Select an IP address out of 65024 at random
- Send a probe querying if address in use, and listen for 2 seconds
  - If positive reply received, restart
  - Otherwise, continue sending probes and listening (2 seconds)
- If K probes sent with no reply, start using the IP number
  - Send 2 packets, at 2 second intervals, asserting IP address is being used
  - If a conflicting assertion received, either:
    - defend (send another asserting packet)
    - defer (stop using the IP address and restart)

Will it work?

- Possible problem…
  - IP number chosen may be already in use, but:
    - Probes or replies may get lost or delayed (host too busy)
- Issues:
  - Self-configuration delays may become unacceptable
    - Would you wait 8 seconds to self-configure your PDA?
  - No justification for parameters
    - for example K=4 in the standard
- Case studies:
  - DTMC and Markov reward models, analytical [BvdSHV03,AK03]
  - TA model using UPPAAL [ZV02]
  - PTA model with digital clocks using PRISM [KNS03]

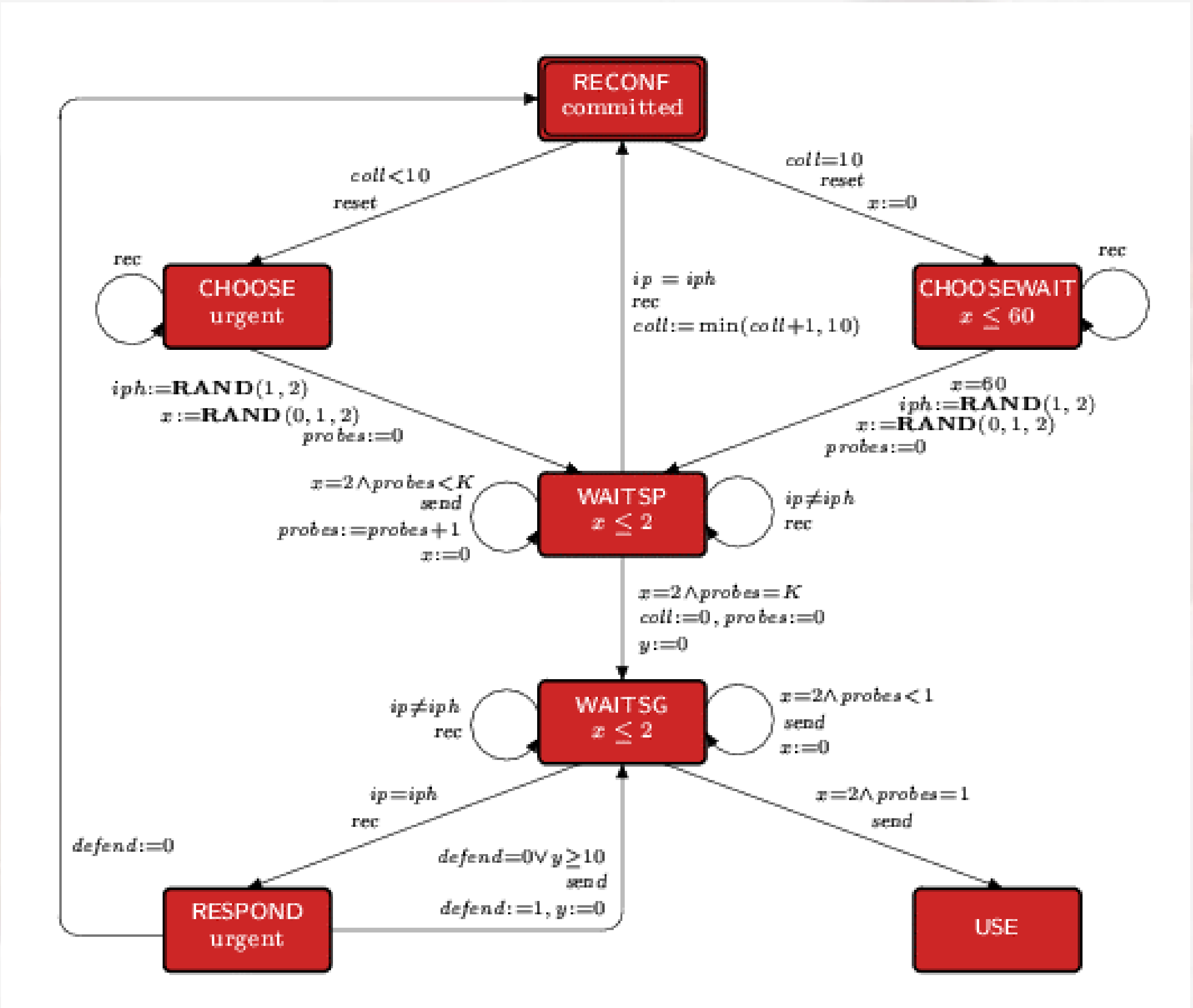
The IPv4 Zeroconf protocol model

- Modelled using Probabilistic Timed Automata (with digital clocks)

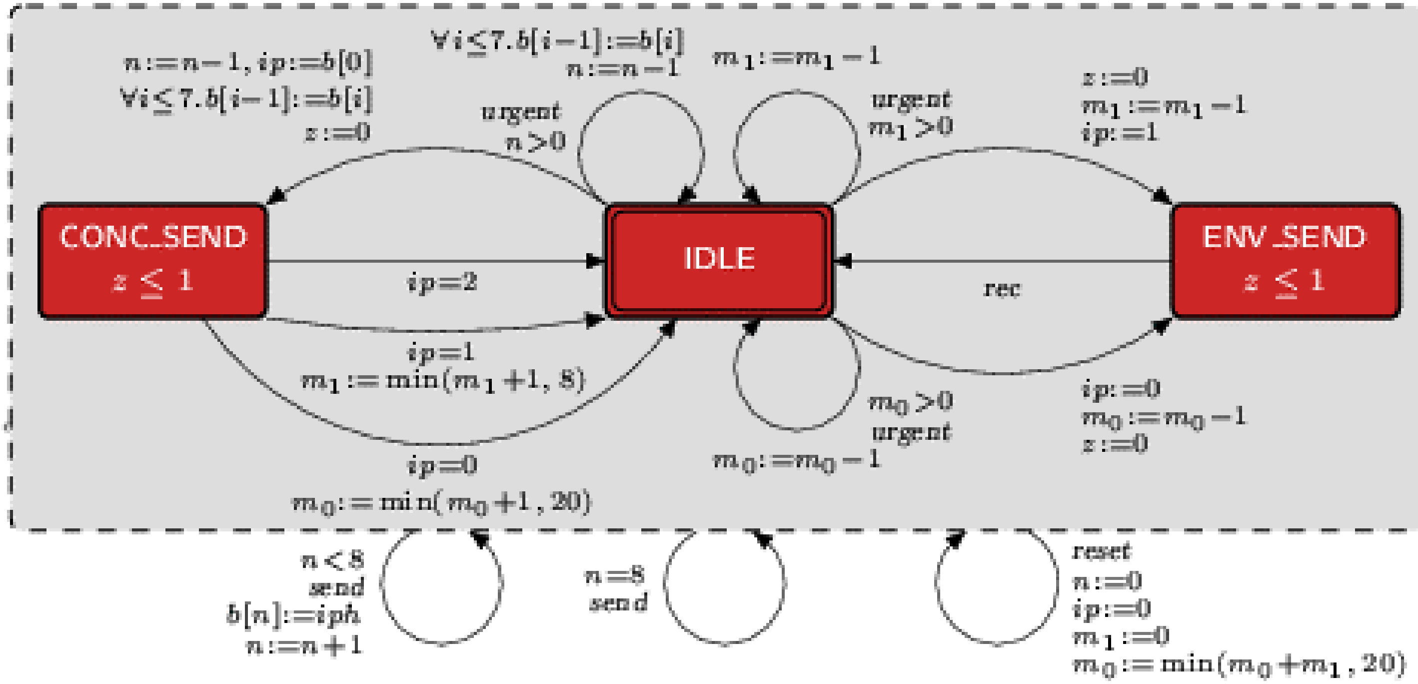
- Parallel composition of two PTAs:
  - one (joining) host, modelled in detail
  - environment (communication medium + other hosts)

- Variables:
  - K (number of probes sent before the IP address is used)
  - the probability of message loss
  - the number of other hosts already in the network

Modelling the host



Modelling the environment



Expected costs

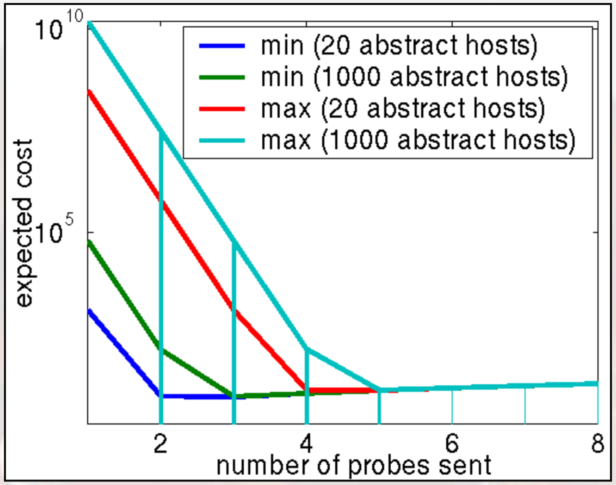
- Compute minimum/maximum expected cost accumulated before obtaining a valid IP address?

- Costs:
  - Time should be costly: the host should obtain a valid IP address as soon as possible
  - Using an IP address that is already in use should be very costly: minimise probability of error

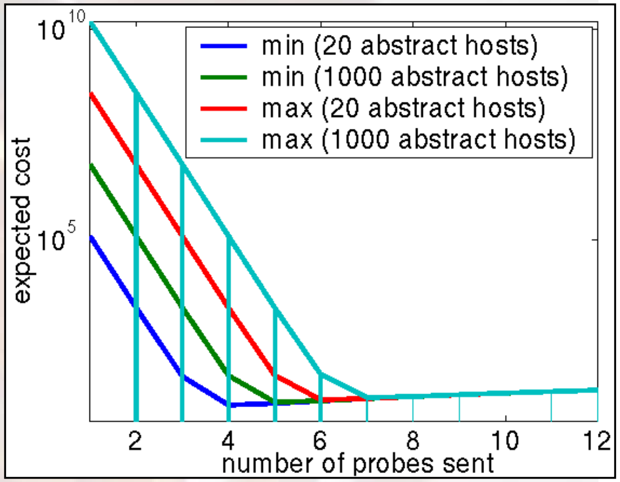
- Cost pair: (r,e)
  - $r=1$ (t time units elapsing corresponds to a cost of t)
  - $e=10^{12}$ for the event corresponding to using an address which is already in use
  - $e=0$ for all other events

Results for IPv4 Zeroconf

Prob. of message loss = 0.001



Prob. of message loss = 0.01



- Sending a high number of probes increases the cost
  - increases delay before a fresh IP address can be used
- Sending a low number of probes increases the cost
  - increases probability of using an IP address already in use
- Similar results to the simpler model of [BvdSHV03]

Successes so far

- Fully automatic, no expert knowledge needed for
  - Probabilistic reachability and temporal logic properties
  - Expected time/cost
- Tangible results!
  - 5 cases of "unusual behaviour" found, over 20 case studies
  - Greater level of detail, may expose obscure dependencies
- PRISM tool robust
  - Simple model description language
  - Broad class of models
  - Large, realistic models often possible
  - Flexible property language
  - Choice of engines

Comparison of model checking engines

- Tandem queueing network
  - "first station becomes fully occupied within t time units"

States:	Time per iteration (sec):		
	MTBDD	Sparse	Hybrid
32,640	0.04	0.05	0.05
130,816	0.06	0.15	0.23
523,776	0.10	0.71	0.99
2,096,128	0.23	-	3.89
33,550,336	0.66	-	-

(450 MHz workstation, 500 MB memory)

Comparison of model checking engines

- Kanban manufacturing system
  - Computation of steady-state probabilities

States:	Time per iteration (sec):		
	MTBDD	Sparse	Hybrid
58,400	41.7	0.04	0.05
454,475	-	0.44	0.50
2,546,432	-	2.76	3.15
11,261,376	-	-	14.8
41,644,800	-	-	58.9

(450 MHz workstation, 1 GB memory)

But…

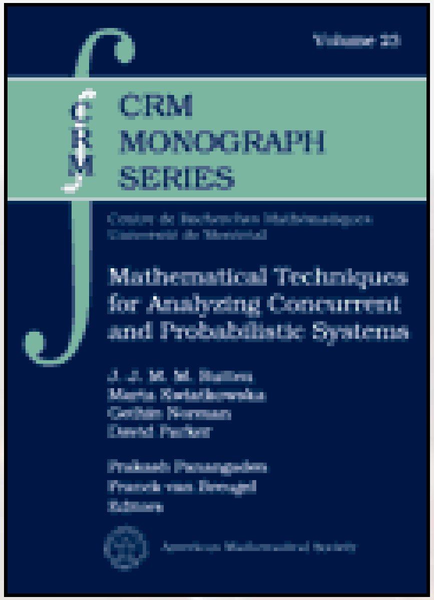
- Models monolithic and finite-state only
  - Emphasis on efficiency
  - No decomposition, abstraction
  - No data reduction
- State-space explosion has not gone away…
  - Heuristics for MTBDDs/BDDs sometimes fail
  - Parallelise? Disk-based?
- Limited expressiveness
  - Only PCTL plus extensions (LTL in progress)
  - Only exponential distributions
  - No direct support for PTAs (work in progress, [FORMATS'04])
  - No continuous space models
  - No mobility

Challenges for future

- Exploiting structure
  - Abstraction, data/equivalence quotient, (de)compositionality…
  - Parametric probabilistic verification?
- Proof assistant for probabilistic verification?
- Approximation methods?
- Efficient methods for continuous models
  - Continuous PTAs? Continuous time MDPs? LMPs?
- More expressive specifications
  - Probabilistic LTL/PCTL*/mu-calculus?
- Real software, not models!

- More applications
  - Quantum cryptographic protocols
  - Mobile ad hoc network protocols

For more information…



J. Rutten, M. Kwiatkowska, G. Norman and D. Parker
[**Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems**]
P. Panangaden and F. van Breugel (editors), CRM Monograph Series, vol. 23, AMS
March 2004

www.cs.bham.ac.uk/~dxp/prism/

- *Case studies, statistics, group publications*
- Download, version 2.1 (900 users)
- Publications by others and courses that feature PRISM…

PRISM Contributors